

CSE102 – Computer Programming

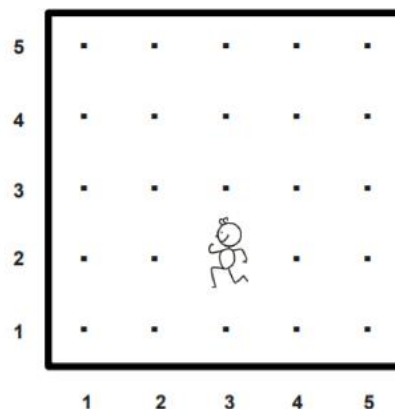
Homework #9

Due Date: 09.05.2020 23:59

Hand in: A student with number 20180000001 should hand in a file named 20180000001.c for this homework and compress it into a .zip file.

Part 1. Stay Home Problem [20pts]

As you know, everyone should stay at home because of the Covid-19 pandemic. Cin Ali, who goes out before the curfew, must run to his home the fastest way before the ban begins.



Let's assume that Cin Ali is in the bakery at the intersection of Street 2 and Avenue 3, as shown in the diagram, and that he wants to return to his home at the intersection of Street 1 and Avenue 1. Even though Cin Ali wants to avoid getting out of the way, there are equally short cuts.

For example, there are three possible ways in this diagram, as follows:

- Move left, then left, then down.
- Move left then down, then left.
- Move down, then left, then left.

Your job in this issue is to write an **recursive function**: [Solutions made without using the recursive function will not be evaluated!]

```
int numPathsHome(int street, int avenue)
```

that returns the number of paths Cin Ali could take back to the origin from the specified starting position, subject to the condition that Cin Ali doesn't want to take any unnecessary steps and can therefore only move west or south (left or down in the diagram).

```
Enter the street number: 2
street:2
Enter the avenue number: 3
avenue:3
Number of optimal paths to take back home: 3
```

Part 2. COVID-19 Health Coverage [50 Points]

Imagine that you are hired as Minister of Health of Turkey. Turkey as well as in the whole world there was a need of a national network of hospitals against the pandemic. As the Minister of Health, you need to investigate possible locations for these new hospitals. Your consultants have prepared a list of potential hospital areas for you.

Each of these hospitals has a cost and will provide coverage to some nearby cities. We can represent each potential hospital area as a **struct** as follows:

```
struct Hospital
{
    char* name;
    char citiesServed[3];
};
```

Although you are interested in providing health care to all, you do not have funds to build an unlimited number of hospitals. Using recursion, you are interested in seeing whether or not it is possible to provide quality health care to every city given a limit on the number of hospitals you can construct. **[Solutions made without using recursion will not be evaluated!]**


Suppose that you are given a char array representing the names of all of the cities in your country. You are also provided with a list of all the proposed hospital locations, each of which is represented by the set of cities that the hospital could service. Given your funding restrictions, you can purchase at most numHospitals total hospitals.

Write a function:

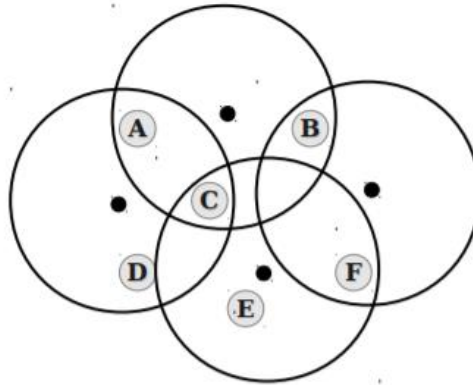
```
int canOfferCovidCoverage(char _cities[6], char _locations[4][3], int
_numHospitals, struct Hospital results[4])
```

that accepts as input the set of all cities and a list of the cities that would be covered by each hospital, along with the maximum number of hospitals that can be constructed, and returns whether or not it is possible to provide coverage to all cities using the limited number of hospitals.

If so, your function should update the result parameter to contain one such choice of hospitals.

 Öyleyse, fonksiyon sonuç parametresini böyle bir hastane seçimini içerecek şekilde güncellemelidir.

As an example, consider the country below, where each city is represented by a letter and each potential hospital location is represented with a black dot:



Here, each hospital is can cover all the cities within their circle of coverage. This would be represented as follows:

- **cities** = { "A", "B", "C", "D", "E", "F" }
- **locations** = { {"A", "B", "C"}, {"A", "C", "D"}, {"B", "F"}, {"C", "E", "F"} }

The topmost hospital would serve cities A, B, and C. The hospital on the left would cover A, C, and D. The hospital on the right covers just B and F, and the hospital on the bottom covers C, E, and F. If you can only purchase two hospitals, then there is no way to guarantee coverage to everyone. However, if you can purchase three hospitals, then you can cover everyone – purchase the top hospital to cover A, B, and C, the bottom hospital to cover C, E, and F, and the leftmost hospital to cover A, C, and D.

```
Enter the maximum number of hospitals that can be constructed:3
```

```
Yes, can offer health care to all!
```

```
Hospital - 1
```

```
Hospital locations: acd
```

```
Hospital - 2
```

```
Hospital locations: bf
```

```
Hospital - 3
```

```
Hospital locations: cef
```

```
Enter the maximum number of hospitals that can be constructed:2
```

```
No, some cities are not covered.
```

Part 3. [30pts]

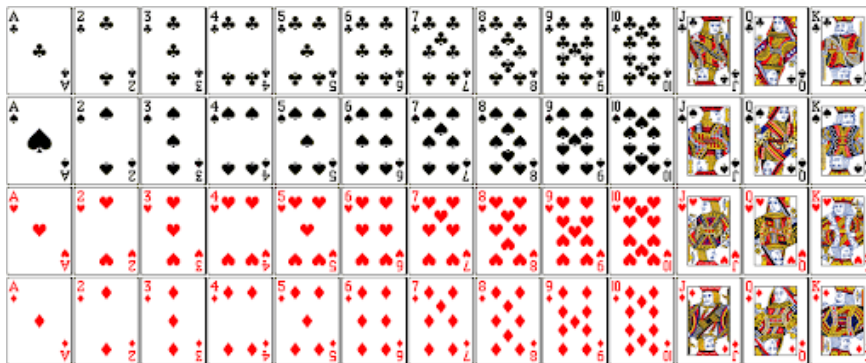
In quarantine days, people get bored more than standing at home and many people spend time with their playing games for fun! In games played with playing cards, it is essential to mix the decks well. Let us mix the cards in the deck with code!

The deck of cards contains 52 cards which have 4 basic suits. These suits:

"Hearts", "Diamonds", "Clubs", "Spades"

There are 13 faces belonging to each suit from 4 suits. These are:

"Ace", "Deuce", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine", "Ten", "Jack", "Queen", "King"



We should represent each card as a **struct** as follows:

```
struct card {
    const char *face;
    const char *suit;
};
```

The steps to be taken are listed below:

- You must place strings in Card structures
- Cards in the deck must be replaced randomly so that the deck is shuffled.
- The elements of the mixed deck should be printed on the screen as in the format below.

```
King of Clubs      Five of Clubs
Deuce of Spades    Five of Diamonds
Four of Clubs      Ace of Spades
Eight of Diamonds  Ten of Clubs
Six of Clubs       Queen of Diamonds
Ace of Clubs       Ten of Hearts
King of Diamonds   Four of Hearts
Seven of Hearts     Five of Hearts
Five of Spades     Eight of Clubs
Ace of Hearts      King of Spades
Deuce of Clubs     Jack of Hearts
Four of Diamonds   Jack of Clubs
Deuce of Hearts    Queen of Clubs
Seven of Spades    Three of Diamonds
Eight of Hearts    King of Hearts
Seven of Clubs     Eight of Spades
Nine of Spades     Nine of Hearts
Six of Hearts      Jack of Diamonds
Nine of Clubs      Jack of Spades
Queen of Hearts    Four of Spades
Three of Spades    Three of Clubs
Seven of Diamonds  Three of Hearts
Nine of Diamonds   Six of Spades
Ten of Diamonds    Ace of Diamonds
Ten of Spades      Six of Diamonds
Queen of Spades    Deuce of Diamonds
```

General Rules:

1. Obey and don't broke the function prototypes that are shown on each part, otherwise, you will get zero from the related part.
2. The program must be developed on Linux based OS and must be compiled with gcc compiler, any problem which rises due to using another OS or compiler won't be tolerated.
3. Note that if any part of your program is not working as expected, then you can get zero from the related part, even it's working in some way.
4. Upload your .zip file on to Moodle to deliver your homework. The zip file must consist of one .c file that contains your solutions. Name format can be found on the top of this homework sheet.
5. You can ask any question about the homework by sending an email to bbuluz@gtu.edu.tr.